คำสั่ง BUFFERS และ FASTOPEN เพื่อการเรียกหาข้อมูลในดิสค์รวดเร็วขึ้น

■ ชัยยงค์ วงศ์ชัยสุวัฒน์

ผู้เขียนเชื่อว่าพวกเราหลาย ๆ ท่านคงรู้จักและเคยใช้คำสั่ง BUFFERS ในการกำหนดจำนวนบัฟเฟอร์ของดิสค์ที่บรรจุอยู่ในไฟล์ CONFIG.SYS มาบ้างแล้ว การติดตั้งและกำหนดจำนวนบัฟเฟอร์ที่พอเหมาะ ซึ่งตามปกติทั่วไปมักนิยมให้มีค่าระหว่าง 10 ถึง 20 เพราะจะเป็นการจัดสรรและสงวนเนื้อที่ของหน่วยความจำไว้ เพื่อเป็นช่องติดต่อนำข้อมูลจากดิสค์เข้ามาเก็บไว้คราวละมาก ๆ โดยไม่ต้องเสียเวลาไปติดต่อกับดิสค์ที่ทำงานช้ากว่าบ่อย ๆ เป็นการเพิ่มความเร็วในการเรียกหาข้อมูลได้อย่างดี

จำนวนบัฟเฟอร์ถ้ากำหนดไว้ไม่เหมาะสมหรือมากเกินไป กลับจะมีผลให้ดอสเสียเวลาค้นหาข้อมูลภายในบัฟเฟอร์ (ถ้าไม่พบจึงไปติดต่อกับดิสค์) และยังทำให้ขนาดของหน่วยความจำเพื่อการใช้งานอื่นลดหายไปมาก ดังนั้นในมุมโอเอสฉบับนี้จึงขอนำรายละเอียดการจัดการและการใช้งานของคำสั่ง BUFFERS มาอธิบาย เพื่อให้ผู้อ่านสามารถเรียกใช้ได้อย่างถูกต้อง และจะอธิบายการเพิ่มขยายคำสั่ง BUFFERS ที่มาพร้อมกับดอส 4 ในการใช้หน่วยความจำส่วนขยายเพิ่มเติม (Expanded Memory) มาขยายจำนวนบัฟเฟอร์ได้สูงถึง 10,000 กับการกำหนดจำนวนบัฟเฟอร์แบบมองล่วงหน้า (look-ahead) สำหรับงานประมวลผลไฟล์แบบเรียงตามลำดับ

คำสั่ง BUFFERS ที่กล่าวข้างต้นจะช่วยงานด้านการอ่านและเขียนข้อมูลภายในไฟล์ของดิสค์ ซึ่งถ้าระบบไฟล์ข้อมูลมีการจัดการเป็นรูปโครงสร้างแบบต้นไม้หลาย ๆ ชั้น การเข้าถึงไฟล์ข้อมูลที่ต้องการ ดอสจะต้องใช้เวลาพอสมควรในการวิ่งเข้าไปค้นหาตามไดเรคทอรีย่อยต่าง ๆ โดยคำสั่ง BUFFERS ไม่ได้ช่วยในเรื่องนี้เลย ดังนั้นในดอส 3.3 จึงได้เพิ่มคำสั่ง FASTOPEN เข้ามาเพื่อสงวนที่หน่วยความจำสำหรับเก็บข้อมูลไดเรคทอรี บอกตำแหน่งของดิสค์ที่ใช้กับไฟล์ข้อมูลที่มีการใช้บ่อย ๆ ทำให้การเรียกค้นข้อมูลที่ผ่านไดเรคทอรีหลาย ๆ ชั้นทำได้เร็วขึ้น นอกจากนี้ในดอส 4 ก็ได้เพิ่มขยายคำสั่ง FASTOPEN ให้สามารถสงวนที่หน่วยความจำสำหรับเก็บข้อมูลคลัสเตอร์บอกผังการจัดสรรดิสค์ของไฟล์ข้อมูล ในมุมโอเอสฉบับนี้จะเสนอรายละเอียดของการเรียกใช้คำสั่ง FASTOPEN เพื่อนำไปใช้ร่วมกับคำสั่ง BUFFERS ในการเพิ่มสมรรถภาพของการเรียกหาข้อมูลในดิสค์ให้รวดเร็วขึ้น

หลักการของ cache

ทั้งคำสั่ง BUFFERS และ FASTOPEN ที่จะกล่าวต่อไปต่างเป็นซอฟต์แวร์ ส่วนที่เรียกว่า cache ที่โปรแกรมดอสมีให้เรียบร้อยแล้ว อาจถือเป็น cache ขั้นพื้นฐานเท่านั้น เพราะปัจจุบันมีซอฟต์แวร์ที่เรียก Disk-caching เพื่อช่วยเร่งความเร็วในการอ่านหรือเขียนข้อมูลกับดิสค์ได้มากกว่า 10 ถึง 30 เท่าออกมาจำหน่ายมากมาย มีทั้งที่ประกอบเสริมในซอฟต์แวร์อื่น เช่น Compaq Disk Cache Utility ที่ให้มาพร้อมกับเครื่องที่มีฮาร์ดดิสค์ IBMCACHE ที่มาพร้อมดอส หรือ SMARTDrive ที่ให้พร้อมโปรแกรม Windows/286, Windows/386 เป็นต้น และที่แยกขายเฉพาะ ราคาตกราว 50 ถึง 100 เหรียญสหรัฐฯ ในที่นี้เราจะมาทำความรู้จักกับคำสั่ง BUFFERS และ FASTOPEN ที่เป็น cache พื้นฐานให้ดีเสียก่อน และถ้าสนใจ cache ระดับสูงขึ้นจึงค่อยหาซอฟต์แวร์เฉพาะมาลองทีหลัง

ซอฟต์แวร์ cache ต่างอาศัยหลักความจริงพื้นฐานสำคัญ 3 ประการคือ

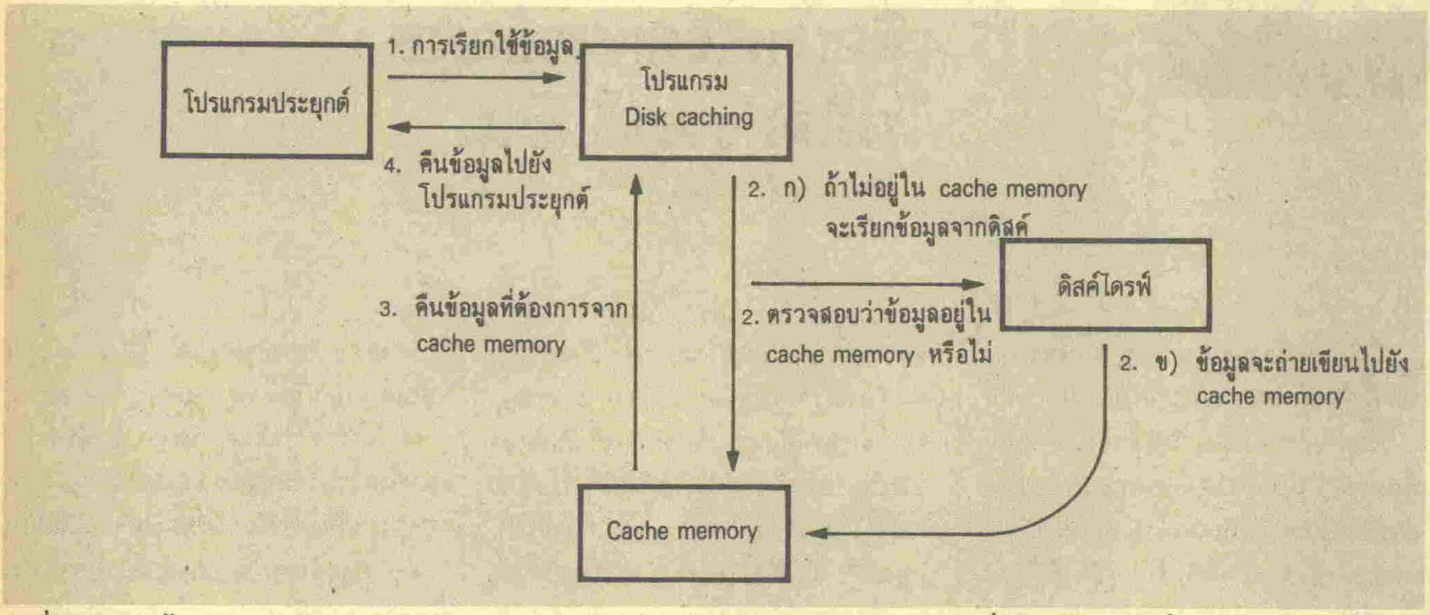
ก. ข้อมูลใดที่มีการเรียกใช้แล้วโอกาสที่จะไปเรียกใช้อีกจะมีค่อนข้างมาก

ข. การจดจำข้อมูลไว้จะดีกว่าที่จะต้องมาเรียกค้นให้เสียเวลา

ค. ข้อมูลที่มีแล้วจะไปช่วยสืบหาข้อมูลที่เกี่ยวข้องข้างเคียงได้ง่าย

จากหลักพื้นฐานดังกล่าว ซอฟต์แวร์ cache จะกันเนื้อที่หน่วยความจำไว้เป็นเนื้อที่เฉพาะที่เรียกว่า cache memory ซึ่งเราสามารถกำหนดขนาดได้เอง เมื่อโปรแกรมมีการเรียกอ่านข้อมูล ข้อมูลจากดิสค์จะส่งไปเก็บที่ cache memory ก่อน แล้วจึงค่อยส่งต่อไปยังโปรแกรมที่เรียกใช้ cache memory จะจดจำข้อมูลเอาไว้ แต่เนื่องจากการอ่านข้อมูลเข้ามาเก็บจะเป็นกลุ่มก้อน ดังนั้นการเรียกใช้ข้อมูลเดิมหรือชุดที่อยู่ใกล้เคียงจึงค้นพบใน cache memory ไม่ต้องไปเรียกจากดิสค์ซึ่งทำงานช้ากว่า

รูปที่ 1 แสดงขั้นตอนการทำงานของซอฟต์แวร์ cache ที่อยู่ระหว่างกลางของโปรแกรมประยุกต์กับดิสค์ไดรฟ์ที่เก็บไฟล์และข้อมูล ในกรณีของคำสั่ง BUFFERS cache memory ก็คือบัฟเฟอร์ของดิสค์ที่จะกล่าวในหัวข้อต่อไปนั่นเอง



รูปที่ 1 แสดงขั้นตอนการเรียกหาข้อมูลจากโปรแกรมประยุกต์ โดยมีโปรแกรม cache คั่นตรงกลาง เพื่อให้ตรวจสอบว่าข้อมูลนั้นอยู่ใน cache memory หรือไม่ ถ้ามีอยู่ โปรแกรม cache จะเรียกข้อมูลจาก cache memory ส่งคืนไปให้โปรแกรมประยุกต์ทันที ถ้าไม่อยู่ก็ต้องเรียกไปยังดิสค์ เพื่อดึงข้อมูลมาเก็บใน cache memory แล้วจึงคืนข้อมูลที่ต้องการไปยังโปรแกรมประยุกต์

การใช้คำสั่ง BUFFERS

คำสั่ง BUFFERS ถือเป็นคำสั่งพิเศษที่ใช้บรรจุในไฟล์จัดระบบ CONFIG.SYS เพื่อกำหนดจำนวนบัฟเฟอร์ของดิสค์ จัดเป็นคำสั่งที่สำคัญมีผลต่อสมรรถภาพการใช้งานดิสค์เป็นอย่างมาก

บัฟเฟอร์ของดิสค์เป็นเนื้อที่สงวนของหน่วยความจำแรม ซึ่งผู้ใช้งานดอสสามารถกำหนดขนาดหรือจำนวนบัฟเฟอร์เอง จุดมุ่งหมายของบัฟเฟอร์ดิสค์ก็เพื่อลดทอนจำนวนครั้งที่ดอสต้องเรียกใช้ข้อมูลจากดิสค์โดยตรงให้มาเรียกใช้ผ่านทางหน่วยความจำแรม เพราะการทำงานเรียกค้นจากดิสค์จะช้ากว่าจากหน่วยความจำแรมมาก การใช้บัฟเฟอร์ดิสค์มีแนวความคิดให้ใช้หน่วยความจำแรมมาก แต่ใช้ดิสค์ไดรฟ์น้อย

ขนาดของบัฟเฟอร์ดิสค์จะสูงกว่าเซกเตอร์ของดิสค์เล็กน้อย ขนาดเซกเตอร์ของดิสค์จะเท่ากับ 512 ไบต์ แต่ขนาดของบัฟเฟอร์ดิสค์จะเท่ากับ 532 ไบต์เป็นอย่างน้อย ดังนั้นขนาดของบัฟเฟอร์หนึ่ง ๆ จึงเก็บข้อมูลจากดิสค์ได้เต็มเซกเตอร์แถมยังจะมีไบต์เพิ่มเติมอีกราว 20 ไบต์สำหรับเก็บข้อมูลเพื่อให้ดอสสืบค้นสิ่งที่เก็บในบัฟเฟอร์ดิสค์ การกำหนดจำนวนบัฟเฟอร์ดิสค์ด้วยคำสั่ง BUFFERS ก็คือการกำหนดขนาดเนื้อที่หน่วยความจำให้สามารถนำข้อมูลจากดิสค์เข้ามาหรือออกไปได้คราวละหลายเซกเตอร์เท่ากับจำนวนบัฟเฟอร์ดิสค์ที่กำหนด

รูปแบบของคำสั่งพิเศษ BUFFERS จะเป็นดังนี้

BUFFERS = nn

โดย nn เป็นจำนวนบัฟเฟอร์ของดิสค์ที่ต้องการ เราสามารถกำหนดจำนวนได้ตั้งแต่ 1 ถึง 99 (ในดอส 4 มีการขยายคำสั่งนี้ออกไปอีก รายละเอียดจะกล่าวถึงในภายหลัง) ถ้าไม่มีการกำหนด ดอสก็จะใช้ค่าดีฟอลต์จาก 2 ถึง 15 บัฟเฟอร์ขึ้นกับชนิดของอุปกรณ์ดิสค์ไดรฟ์ ขนาดของหน่วยความจำ และเวอร์ชันของดอสเอง ตารางที่ 1 แสดงค่าดีฟอลต์ของจำนวนบัฟเฟอร์ดิสค์แยกตามเวอร์ชันและฮาร์ดแวร์

ตารางที่ 1 ค่าดีฟอลต์ของจำนวนบัฟเฟอร์ดิสค์

ดอสเวอร์ชัน	ฮาร์ดแวร์	จำนวนบัฟเฟอร์
ก่อนดอส 3.3	ดิสค์ไดรฟ์จานอ่อน (floppy)	2
	ดิสค์ไดรฟ์จานแข็ง (fixed)	3
ดอส 3.3 เป็นต้นไป	360 KB ดิสค์ไดรฟ์จานอ่อน	2
	ดิสค์ไดรฟ์ชนิดอื่น ๆ มากกว่า 360 KB	3
	หน่วยความจำมากกว่า 128 KB	5
	หน่วยความจำมากกว่า 256 KB	10
	หน่วยความจำมากกว่า 512 KB	15

เราอาจกำหนดจำนวนบัฟเฟอร์น้อยกว่าค่าดีฟอลต์ แต่โดยปกติแล้วเราจะนิยมกำหนดค่าให้สูงกว่าค่าดีฟอลต์เป็นส่วนใหญ่

ดอสใช้บัฟเฟอร์ดิสค์อย่างไร

ดอสจะอ่านข้อมูลจากดิสค์ หรือนำข้อมูลไปเก็บในดิสค์เป็นจำนวนเต็มทั้งเซกเตอร์ ดอสไม่สามารถอ่านหรือเขียนข้อมูลติดต่อกับดิสค์เป็นเศษของเซกเตอร์ ถึงแม้ว่าข้อมูลที่ต้องการจริงไม่เต็มตามเซกเตอร์ของดิสค์ (ไม่เป็นสัดส่วนของ 512 ไบต์)

เมื่อโปรแกรมต้องการเรียกข้อมูลไม่ครบเต็มขนาดของเซกเตอร์ ดอสจะอ่านข้อมูลเต็มขนาดทั้งเซกเตอร์เข้ามาเก็บในบัฟเฟอร์ดิสค์ หากโปรแกรมต้องการข้อมูลต่อไปอีก ข้อมูลที่ต้องการใหม่มีแนวโน้มว่าจะอยู่ในบัฟเฟอร์ดิสค์เรียบร้อยแล้ว ดังนั้นการเรียกค้นข้อมูลจะทำบนหน่วยความจำแรมของบัฟเฟอร์ที่รวดเร็วกว่าแทนที่จะต้องไปเรียกค้นในดิสค์ที่ค่อนข้างช้ากว่ามาก แต่ถ้าข้อมูลใหม่ที่ต้องการไม่อยู่ในบัฟเฟอร์ดิสค์ ดอสก็จะสวิตช์ไปเรียกค้นในดิสค์เหมือนเดิม

ในทำนองเดียวกันการเขียนบันทึกข้อมูลลงสู่ดิสค์ หากข้อมูลนั้นไม่ครบเต็มเซกเตอร์ ดอสจะรวบรวมข้อมูลไว้ในบัฟเฟอร์ดิสค์ก่อน และเมื่อบัฟเฟอร์เต็มจึงค่อยมีการถ่ายเทข้อมูลจากบัฟเฟอร์ลงสู่ดิสค์ต่อไป กรณีการสั่งปิดไฟล์ข้อมูลที่เก็บในบัฟเฟอร์ดิสค์ก็จะมีการถ่ายเทไปยังไฟล์ในดิสค์ก่อนเสมอ

เนื่องจากจำนวนบัฟเฟอร์ดิสค์ที่เรากำหนดขึ้นมีจำนวนจำกัด และงานที่ต้องการใช้ดิสค์ก็มีอยู่มากมาย ดอสจึงหาวิธีการหรือขั้นตอนมาตรวจสอบการใช้งานของบัฟเฟอร์ เพื่อให้มีการสลับหมุนเวียนบัฟเฟอร์สำหรับเรียกติดต่อข้อมูลกับดิสค์ โดยบัฟเฟอร์แต่ละตัวที่กำหนดขึ้นจะมีข้อมูลกำกับว่ามีการเรียกใช้เมื่อเร็ว ๆ นี้หรือไม่ เมื่อมีการเรียกติดต่อกับดิสค์เพื่อนำข้อมูลเข้ามาในบัฟเฟอร์ดิสค์ ดอสจะต้องตรวจสอบไปที่รายการของบัฟเฟอร์ว่า บัฟเฟอร์ใดที่ไม่มีการเรียกใช้เป็นเวลานานแล้ว ข้อมูลเซกเตอร์ใหม่จากดิสค์จะเข้ามาแทนในบัฟเฟอร์นั้นวิธีนี้เรียกว่า ***การเรียกใช้เมื่อเร็ว ๆ นี้น้อยที่สุด*** (least recently used-LRU) วิธีการ LRU นี้จะช่วยให้โปรแกรมและดอสทำงานได้รวดเร็วขึ้น

เมื่อมีการติดตั้ง BUFFERS กำหนดบัฟเฟอร์ดิสค์ ดอสจะจัดการงานต่าง ๆ ที่ติดต่อกับดิสค์ให้เอง เมื่อเรารันโปรแกรมประยุกต์ใช้งานแล้วมีการอ่านเรียกไฟล์ข้อมูล ดอสจะนำส่วนของไฟล์เข้าไปเก็บในบัฟเฟอร์ของดิสค์ที่เป็นหน่วยความจำแรม และเมื่อมีการบันทึกข้อมูลลงสู่ดิสค์ ข้อมูลนั้นจะไปที่บัฟเฟอร์ดิสค์ก่อน บัฟเฟอร์ดิสค์สามารถใช้กับไฟล์ทุกชนิด รวมถึงไฟล์โปรแกรมเสริมที่เราเรียกว่า โอเวอร์เลย์ (Overlays) ดอสจะโหลดโอเวอร์เลย์ไฟล์ลงในบัฟเฟอร์ดิสค์ แล้วผ่านไปให้โปรแกรมหลักต่อไป

โปรแกรมที่มีการเรียกใช้ดิสค์ด้วยการอ่านหรือเขียนข้อมูลไปยังบริเวณของไฟล์ที่แยกส่วนกัน (หรือ random) เช่น การเรียกค้นไฟล์ดาต้าเบสอย่างมีเงื่อนไขสืบค้นจำเป็นจะต้องกำหนดจำนวนบัฟเฟอร์สูงกว่าปกติ เพราะถ้าจำนวนบัฟเฟอร์มากแล้วโอกาสของข้อมูลที่ต้องการจะอยู่ในบัฟเฟอร์ดิสค์จะมีมาก ทำให้การทำงานรวดเร็วขึ้น โปรแกรมที่มีการเรียกใช้ดิสค์ด้วยการอ่านหรือเขียนข้อมูลแบบเรียงลำดับ (Sequential) โดยเริ่มจากต้นไฟล์เรียงไปจนจบไฟล์ ในกรณีนี้เราจะไม่ได้ประโยชน์จากบัฟเฟอร์ของดิสค์นัก เนื่องจากการนำข้อมูลมาเก็บในบัฟเฟอร์ไม่ได้เกิดจากการเรียงลำดับของบัฟเฟอร์ แต่จะเกิดจากการใช้วิธีการ LRU ตามที่อธิบายมาแล้ว

กล่าวโดยสรุปการกำหนดจำนวนบัฟเฟอร์ดิสค์จะให้ประโยชน์สำหรับการอ่านหรือเขียนข้อมูลที่มีขนาดไม่เท่ากับขนาดเซกเตอร์ของดิสค์ และจะใช้งานได้ดีสำหรับกรณีที่มีการอ่านหรือเขียนแบบสุ่มหรือ random ในดอส 4 ได้เพิ่มวิธีการกำหนดบัฟเฟอร์ดิสค์แบบมองล่วงหน้าที่เหมาะสำหรับการอ่านหรือเขียนแบบเรียงตามลำดับ (Sequential) ซึ่งจะได้กล่าวถึงต่อไป

การประมาณจำนวนของบัฟเฟอร์ที่เหมาะสม

โดยทั่วไปผู้ใช้งานดอสหรือโปรแกรมประยุกต์ต่าง ๆ จะเป็นผู้กำหนดจำนวนบัฟเฟอร์ดิสค์ขึ้นเอง เพื่อให้เหมาะสมกับโปรแกรมที่ใช้และขนาดของหน่วยความจำที่มีอยู่ ดังนั้นจึงเป็นการยากที่จะกำหนดหรือตั้งตัวเลขที่แน่นอน ในสภาพปกติแล้วจำนวนบัฟเฟอร์ดิสค์มักจะกำหนดให้สูงราว 10-20 บัฟเฟอร์ และถ้านำไปใช้ในงานบัญชีหรืองานฐานข้อมูล จำนวนบัฟเฟอร์ก็จะสูงขึ้นไปอีก

หน่วยความจำของระบบที่มีอยู่จะเป็นตัวช่วยกำหนดจำนวนบัฟเฟอร์ เพราะบัฟเฟอร์จะใช้เนื้อที่หน่วยความจำไปส่วนหนึ่ง ทำให้หน่วยความจำสำหรับโปรแกรมลดน้อยลง ถ้าระบบนั้นมีหน่วยความจำแรมน้อย เช่น เพียง 256 กิโลไบต์ แต่กำหนดบัฟเฟอร์สูง ๆ โอกาสที่จะไปรันโปรแกรมขนาดใหญ่ย่อมทำไม่ได้อย่างแน่นอน การกำหนดจำนวนบัฟเฟอร์มาก ๆ นอกจากทำให้หน่วยความจำหลักสำหรับ การรันโปรแกรมน้อยลงแล้ว ยังมีผลทำให้ระบบทำงานช้าลง เพราะดอสจะต้องใช้เวลาพอสมควรเข้าไปค้นหาข้อมูลที่ต้องการอ่านหรือเขียนภายในบัฟเฟอร์ที่มีอยู่จำนวนมาก

เมื่อบัฟเฟอร์ดิสค์จำเป็นต้องมีปริมาณพอเหมาะกับงานและระบบที่มีอยู่ การกำหนดจำนวนบัฟเฟอร์จึงควรมีการปรับแต่งเฉพาะสำหรับงานหรือระบบ ข้อเสนอแนะสำหรับการเริ่มต้น เราอาจเริ่มที่ BUFFERS = 10 สำหรับเครื่องระบบดิสเกตต์

หรือ BUFFERS = 20 สำหรับเครื่องระบบฮาร์ดดิสก์ ลองวัดหรือตรวจสอบสมรรถภาพการใช้งานว่าช้าเร็วขนาดไหน จากนั้นค่อยทำการปรับเพิ่มหรือลดจำนวนบัฟเฟอร์ระหว่าง 1 ถึง 5 บัฟเฟอร์ บูตระบบใหม่และวัดสมรรถภาพอย่างสม่ำเสมอทุกวันจนกระทั่งได้สมรรถภาพที่ดีที่สุด ซึ่งการตรวจสอบเราไม่ต้องวัดค่ากันอย่างละเอียดเกินไปนัก เพียงทำอย่างหยาบ ๆ หรืออาศัยการสังเกตก็รู้สึกได้

การเพิ่มเติมคำสั่ง BUFFERS ในดอส 4

ในดอส 4 ได้มีการขยายเพิ่มเติมคำสั่ง BUFFERS ออกไปโดยมีรูปแบบคำสั่งดังนี้

BUFFERS = nn, l /x

ส่วนแรกที่เพิ่มเติมในดอส 4 คือการนำหน่วยความจำส่วนขยายเพิ่มเติม (Expanded Memory) มาเสริมให้สามารถกำหนดจำนวนบัฟเฟอร์ได้สูงสุดถึง 10,000 ซึ่งเป็นการเพิ่มสมรรถภาพที่ดีสำหรับโปรแกรมขนาดใหญ่และไฟล์ข้อมูลแบบสุ่ม ส่วนเพิ่มเติมที่สองคือการกำหนดบัฟเฟอร์แบบมองล่วงหน้า (look-ahead) ได้สูงสุดถึง 8 เป็นกลไกที่ช่วยงานด้านการประมวลไฟล์แบบเรียงลำดับ

ตัวอย่างคำสั่ง BUFFERS

BUFFERS = 15

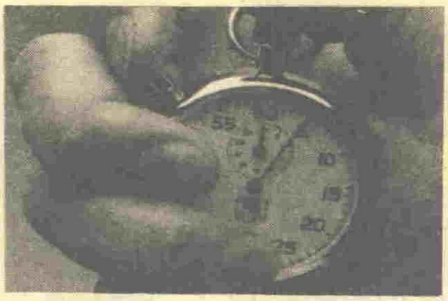
เป็นการกำหนดจำนวนบัฟเฟอร์ดิสก์เท่ากับ 15 (เราสามารถกำหนดได้จาก 1 ถึง 99)

BUFFERS = 15, 3

เป็นการกำหนดจำนวนบัฟเฟอร์ดิสก์เท่ากับ 15 และจำนวนบัฟเฟอร์แบบมองล่วงหน้าเท่ากับ 3

BUFFERS = 500, 3

เป็นการกำหนดบัฟเฟอร์ดิสก์เท่ากับ 500 จำนวนบัฟเฟอร์แบบมองล่วงหน้าเท่ากับ 3 ในหน่วยความจำเพิ่มขยาย



(Expanded memory) ที่แสดงด้วยสวิตช์/X ภายในคำสั่ง

เหตุที่เราสามารถเพิ่มขยายจำนวนบัฟเฟอร์ดิสก์สูงขึ้นได้นั้น เนื่องมาจากในดอส 4 สามารถใช้งานการเพิ่มขยายหน่วยความจำ LIM/EMS 4 (ระบบการเพิ่มขยายหน่วยความจำตามมาตรฐานของ Lotus, Intel และ Microsoft) ได้อย่างเต็มที่ ซึ่งจะประกอบด้วยดีไวซ์ไดรเวอร์ EMS และฮาร์ดแวร์การ์ด EMS โดยไดรเวอร์ EMS จะเชื่อมกรอบหน้าขนาด 16 กิโลไบต์ในบริเวณหน่วยความจำที่ไม่ใช้งานจากตำแหน่งของหน่วยความจำ 640 กิโลไบต์ถึง 1 เมกะไบต์ ไปยังขอบหน้าของหน่วยความจำเพิ่มขยายขนาด 16 กิโลไบต์

ในดอสเวอร์ชันเก่าบัฟเฟอร์ดิสก์จะเป็นส่วนของหน่วยความจำปกติภายใน 640 กิโลไบต์ แต่ในดอส 4 เมื่อกำหนดด้วยสวิตช์ /x จะเป็นการสั่งงานให้ใช้บัฟเฟอร์ในหน่วยความจำเพิ่มขยาย ซึ่งสามารถกำหนดได้สูงถึง 10,000 บัฟเฟอร์หรือประมาณเท่ากับ 5 เมกะไบต์ การโยงไปยังบัฟเฟอร์หรือหน่วยความจำเพิ่มขยายจะจัดการผ่านทางกรอบหน้าขนาด 16 กิโลไบต์ (ที่กล่าวถึงข้างต้น) เรียกว่ากรอบหน้าหมายเลข 255 ขณะเดียวกันก็จะมีวิธีการจัดการบัฟเฟอร์ด้วยการค้นหาจาก hash table ที่สร้างในหน่วยความจำปกติของ 640 กิโลไบต์

แต่ละรายการของ hash table จะชี้ไปยังบัฟเฟอร์ดิสก์ได้แน่นอนเพียง 15 บัฟเฟอร์ โดยที่แต่ละบัฟเฟอร์ดิสก์มีขนาด 532 ไบต์ ดังนั้นขอบหน้าขนาด 16 กิโลไบต์จึงบรรจุบัฟเฟอร์ดิสก์ได้เพียง 30 บัฟเฟอร์ เป็นการกำหนดรายการ hash เพียง 2 รายการ ตัวอย่างของการใช้คำสั่ง BUFFERS =50 /x ใน CONFIG.SYS จะเป็นการสร้างรายการ hash จำนวน 4 รายการในหน่วยความจำปกติ และมีบัฟเฟอร์ดิสก์ได้เท่ากับ 60 ในหน่วยความจำเพิ่มขยายภายในขอบหน้าขนาด 16 กิโลไบต์ 2 หน้า

ในรายการของ hash table หนึ่ง ๆ จะแบ่งได้เป็น 3 ฟิลด์ ฟิลด์แรกเรียกว่า EMS page เป็นหมายเลขขอบหน้าขนาด 16 กิโลไบต์ สำหรับให้ EMS จัดการเพื่อโยงไปยังกรอบหน้าหมายเลข 255 ฟิลด์ที่สองชื่อ Buffer Head ขนาด 4 ไบต์ที่เป็นตัวชี้ระบุไปยังบัฟเฟอร์แรกของลิสต์วงรอบ (circular list) ที่ประกอบด้วยบัฟเฟอร์ดิสก์จำนวน 15 บัฟเฟอร์ ฟิลด์สุดท้ายคือ Dirty Count เป็นตัวบอกจำนวนบัฟเฟอร์ที่มีการเปลี่ยนแปลง (ดูรูปที่ 2 ประกอบ)

ในการจัดการลิสต์วงรอบ ดอสจะใช้วิธีการ LRU เมื่อต้องการใช้บัฟเฟอร์แต่ไม่มีบัฟเฟอร์ว่าง ดอสจะย้ายบัฟเฟอร์ที่ไม่ค่อยอ้างถึงเป็นเวลานานแล้วออกจากลิสต์ ขั้นตอนการอ่านเซกเตอร์ข้อมูลจากดิสก์จะปฏิบัติดังนี้

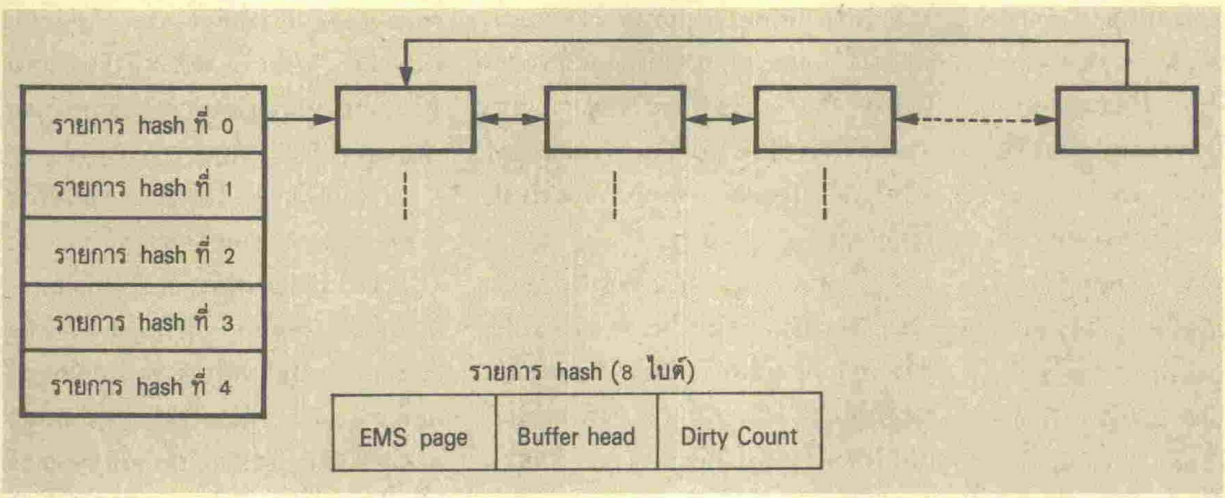
ก. คำนวณ N = เศษของผลหาร (หมายเลขเซกเตอร์/จำนวนรายการ hash)

ข. โยง EMS page ของรายการ hash หมายเลข N ไปยังกรอบหน้าหมายเลข 255 ถ้ามีสวิตช์ /x ระบุไว้

ค. ค้นในลิสต์วงรอบสำหรับเซกเตอร์ที่ต้องการในรายการ hash หมายเลข N

ง. ถ้าพบเซกเตอร์ที่ต้องการให้กำกับว่าบัฟเฟอร์นั้นถูกใช้เร็ว ๆ นี้มากที่สุด แล้วคืนกลับมาอ่านข้อมูลจากหน่วยความจำแทนการอ่านจากดิสก์

จ. ถ้าไม่พบให้ค้นหาบัฟเฟอร์ที่ถูกใช้เร็ว ๆ นี้น้อยที่สุด ถ้าบัฟเฟอร์นั้นมีการเปลี่ยนแปลงค่าก็เก็บบันทึกในดิสก์ก่อน จากนั้นปลดให้บัฟเฟอร์ว่าง



รูปที่ 2 โครงสร้างของ hash table ภายในหน่วยความจำปกติ

ฉ. อ่านข้อมูลจากเซกเตอร์ที่ต้องการเข้ามาในบัฟเฟอร์ว่าง แล้วกำกับว่าเพิ่งมีการใช้เร็ว ๆ นี้มากที่สุด

ช. คืนกลับให้ข้อมูลจากเซกเตอร์ผ่านบัฟเฟอร์ของหน่วยความจำ

เมื่อเราใช้คำสั่ง BUFFERS ที่มีสวิตช์ /X อยู่ด้วย กรอบหน้าหมายเลข 255 ได้ถูกกำหนดใช้งานไปแล้ว ดังนั้นในการเขียนโปรแกรมประยุกต์ใด ๆ จะต้องละเว้นการใช้งานในบริเวณกรอบหน้าหมายเลข 255 ขนาด 16 กิโลไบต์นั้น

บัฟเฟอร์ดิสค์แบบมองล่วงหน้า

บัฟเฟอร์แบบมองล่วงหน้าเป็นเนื้อที่หน่วยความจำปกติที่มีขนาดเป็น 512 ไบต์ ซึ่งมีการกำหนดค่าได้จาก 1 ถึง 8 หรือสงวนที่จาก 0.5 กิโลไบต์ถึง 4 กิโลไบต์ จุดมุ่งหมายของบัฟเฟอร์แบบมองล่วงหน้าจะใช้อ่านข้อมูลเซกเตอร์ถัด ๆ ไปจากเซกเตอร์ที่อ้างถึงในไฟล์แบบเรียงลำดับเข้ามาล่วงหน้า ทำให้จำนวนครั้งของการอ่านข้อมูลจากดิสค์น้อยลง

เมื่อมีการกำหนดบัฟเฟอร์แบบมองล่วงหน้าเป็น 3 การอ่านข้อมูลเซกเตอร์ที่ 210 จากดิสค์จะเป็นการอ่านข้อมูลในเซกเตอร์ 210, 211 และ 212 เข้ามาพร้อมกัน ถ้าดอสต้องการอ่านข้อมูลในเซกเตอร์ที่ 211 ในการเรียกใช้ครั้งต่อไปก็จะนำมาจากในบัฟเฟอร์มองล่วงหน้า ไม่ต้องไปค้นหาในดิสค์ ดอสมีวิธีการตรวจสอบว่ากรณีใดควรค้นจากบัฟเฟอร์ดิสค์ปกติของการสุ่มหรือค้นจากบัฟเฟอร์มองล่วงหน้าของการเรียงลำดับด้วยกฎเกณฑ์ง่าย ๆ ของการเปรียบเทียบ กล่าวคือ ถ้า C เป็นหมายเลขเซกเตอร์ที่ต้องการข้อมูลปัจจุบันและ P เป็นหมายเลขเซกเตอร์ก่อนหน้านี้ ให้ดูผลต่างของ C-P ถ้ามีค่าน้อยกว่าหรือเท่ากับ 1 ดอสจะอ่านค้นในบัฟเฟอร์ล่วงหน้า ไม่เช่นนั้นดอสจะเลือกค้นเฉพาะเซกเตอร์ C

การกำหนดบัฟเฟอร์แบบล่วงหน้าช่วยเพิ่มสมรรถภาพในการเรียกใช้ข้อมูลแบบเรียงลำดับได้ดี ยิ่งถ้าเป็นไฟล์ขนาดใหญ่และจำนวนบัฟเฟอร์ล่วงหน้ากำหนดไว้สูง การอ่านข้อมูลจะเร็วขึ้นมากทีเดียว

การใช้คำสั่ง FASTOPEN

คำสั่ง FASTOPEN เป็นคำสั่งใหม่ที่เพิ่งเข้ามาในดอส 3.3 เพื่อใช้งานกับฮาร์ดดิสค์เท่านั้น คำสั่งนี้จะช่วยเสริมงานที่คำสั่ง BUFFERS ยังมีช่องว่างอยู่ ช่วยให้เราสามารถเปิดไฟล์ได้อย่างรวดเร็ว แต่ละไฟล์ที่เก็บในดิสค์จะมีข้อมูล 32 ไบต์ที่เกี่ยวกับไฟล์ ได้แก่ ชื่อ นามสกุลไฟล์ แอตทริบิวต์ วันที่ เวลาที่สร้าง ขนาด และตำแหน่งเริ่มต้นของไฟล์ในดิสค์มารวมเก็บเป็นไดเรคทอรี

เมื่อมีคำสั่งให้เปิดไฟล์ ดอสจะอ่านไดเรคทอรีของไฟล์จากเซกเตอร์ของดิสค์ลงสู่บัฟเฟอร์ดิสค์ เพื่อว่าในการอ่านหรือเขียนคราวต่อไปจะใช้ข้อมูลนั้นได้ แต่เนื่องจากในการประยุกต์ใช้งานจะมีการเปิดอ่านเขียน และปิดไฟล์เดิมอยู่เสมอ ทำให้เกิดความล่าช้าของการทำงาน นั่นคือ ถ้ามีการเรียกเปิดไฟล์ที่อยู่ในไดเรคทอรีย่อยลึกหลาย ๆ ชั้นของโครงสร้างต้นไม้ ดอสจะต้องเรียกไดเรคทอรีย่อยเข้ามาหลายชั้นจนไปถึงข้อมูลของไฟล์ ทำให้ต้องเสียเวลาและเปลืองเนื้อที่ของบัฟเฟอร์ นอกจากนี้เนื่องจากบัฟเฟอร์ดิสค์มีจำนวนจำกัดจึงอาจมีโอกาสที่ต้องถ่ายเทข้อมูลออกจากบัฟเฟอร์แล้วอ่านข้อมูลเข้ามาใหม่ เมื่อจะทำงานจัดการในไฟล์ใหม่ก็ต้องเรียกไดเรคทอรีเข้ามาใหม่

คำสั่ง FASTOPEN ใช้หลักการ cache ช่วยแก้ปัญหาข้างต้น โดยจะใช้เนื้อที่หน่วยความจำนำข้อมูลไดเรคทอรีของไฟล์ที่เปิดหรือเรียกใช้บ่อย ๆ มาเก็บเอาไว้ ซึ่งข้อมูลส่วนหนึ่งจะบอกที่เก็บของไฟล์ในดิสค์ จากข้อมูลนี้ก็โยงไปยังผังจัดวางไฟล์ในดิสค์ (File Allocation Table-FAT) ที่จะอ่าน

เข้ามาในบัฟเฟอร์ให้ติดตามบอกลำดับการเก็บข้อมูลไฟล์ในดิสก์เรียงจากต้นไปจนจบไฟล์ เมื่อมีการเรียกไฟล์เดิม ข้อมูลไดเรคทอรีจะคงอยู่ในหน่วยความจำ ทำให้ไม่ต้องอ่านจากดิสก์หลาย ๆ ครั้งอีก

รูปแบบคำสั่งของ FASTOPEN คือ

FASTOPEN d: = nnn...

โดย FASTOPEN เป็นชื่อของโปรแกรมนามสกุล .EXE อยู่ในดอสตั้งแต่เวอร์ชัน 3.3 เป็นต้นไป

d: คือดิสก์ไดรฟ์ที่เป็นฮาร์ดดิสก์ที่ต้องการเก็บรายการไดเรคทอรีในหน่วยความจำ

nnn เป็นจำนวนรายการไดเรคทอรี โดยแต่ละไฟล์หรือไดเรคทอรีย่อยจะเรียกเป็นหนึ่งรายการ ซึ่งสามารถตั้งค่าได้จาก 10 ถึง 999 แต่ค่าดีฟอลต์คือ 34 ต่อไดรฟ์

ถ้าหากมีฮาร์ดดิสก์มากกว่าหนึ่งไดรฟ์ เราสามารถกำหนดไดรฟ์ใหม่ต่อได้ทันทีตามสัญลักษณ์...ที่เขียนต่อท้ายคำสั่ง แต่ผลรวมของรายการไฟล์หรือไดเรคทอรีย่อยที่ใช้ในทุกไดรฟ์จะต้องไม่เกิน 999 ในทางปฏิบัติทั่วไปแล้วค่าของรายการไดเรคทอรีจะจำกัดอยู่ระหว่าง 100 ถึง 200 ต่อดิสก์ไดรฟ์ เพราะหากกำหนดไว้สูงเกินไป ดอสจะต้องวนเวียนค้นอยู่ในรายการไดเรคทอรีภายใน ซึ่งอาจช้ากว่าการอ่านข้อมูลจากดิสก์โดยตรง และเนื่องจากข้อมูลไดเรคทอรีแต่ละรายการใช้เนื้อที่หน่วยความจำรวม 35 ไบต์ การกำหนดค่ารายการไดเรคทอรีระหว่าง 100 ถึง 200 จึงเป็นค่าที่พอเหมาะไม่ใช้หน่วยความจำมากเกินไป

ขณะเดียวกันการกำหนดรายการไดเรคทอรีก็ต้องไม่ให้น้อยเกินไป เพราะจะทำให้ต้องแย่งกันใช้เนื้อที่หน่วยความจำที่จองไว้ เช่นเดียวกับกรณีคำสั่ง BUFFERS วิธีการ LRU จะใช้เพื่อตรวจสอบยกเลิกรายการไดเรคทอรีที่ถูกใช้เร็ว ๆ นี้น้อยที่สุดหรือไม่ได้ใช้นานแล้ว เพื่อนำรายการใหม่เข้าไปแทน

ข้อควรระวังของการใช้คำสั่ง FASTOPEN นี้จะต้องไม่กำหนดช่องไดรฟ์ที่ผ่าน การเปลี่ยนทิศหรือหลอกไว้ของคำสั่ง JOIN, SUBST หรือ ASSIGN ทั้งสามคำสั่งเป็นคำสั่งแสร้งที่ทำให้การใช้งาน FASTOPEN ผิดพลาด (คำสั่ง ASSIGN ใช้เพื่อกำหนดช่องดิสก์ไดรฟ์หนึ่งไปเป็นอีกช่องหนึ่ง คำสั่ง SUBST เป็นการกำหนดไดเรคทอรีย่อยหลายชั้นเหลือเพียงช่องไดรฟ์หนึ่ง และคำสั่ง JOIN เชื่อมให้มองดิสก์ช่องหนึ่งเป็นไดเรคทอรีหนึ่ง)

เช่นเดียวกับกรณีการกำหนดจำนวนบัฟเฟอร์ดิสก์ เราไม่สามารถระบุตัวเลขของจำนวนรายการไดเรคทอรีที่เหมาะสมแน่นอนได้ เพราะในแต่ละระบบงานค่าจะแตกต่างกัน ดังนั้นเราจะต้องใช้วิธีตรวจสอบสมรรถภาพด้วยการลองค่า ซึ่งอาจเริ่มต้นที่ค่า 100 แล้วปรับเปลี่ยนค่ามากขึ้นหรือลดลงทีละ 5 จนเห็นว่าค่าใดดีที่สุดก็ใช้ค่านั้น

การขยายคำสั่ง FASTOPEN ในดอส 4

นอกจากจะมีการปรับปรุงคำสั่ง BUFFERS ในดอส 4 แล้ว คำสั่ง FASTOPEN ก็ได้รับการปรับปรุง ด้วยการเพิ่มกลไกที่เรียกว่า Fast Seek หรือการค้นหาอย่างฉับไวของการนำข้อมูลคลัสเตอร์ของไฟล์มาเก็บในหน่วยความจำที่สงวนไว้ ทำให้การเรียกเข้าถึงไฟล์เพื่อการอ่านและเขียนทำได้รวดเร็วขึ้น เมื่อเปรียบเทียบกับข้อมูลคลัสเตอร์ของไฟล์ที่เก็บในตารางหรือผังจัดวางไฟล์ (FAT) ของดิสก์แล้ว หลักการ cache ของรายการคลัสเตอร์ของไฟล์แบบ Fast Seek จะดีกว่า เพราะมีวิธีการแทนข้อมูลที่หนาแน่นจุได้มาก และยังสามารถเพิ่มขยายให้โตขึ้นเรื่อย ๆ

หลักการของ Fast Seek จะเป็นการเก็บหมายเลขคลัสเตอร์ของไฟล์ในหน่วยความจำ ในแต่ละไฟล์จะเก็บในหนึ่งคลัสเตอร์ถ้าเป็นไฟล์เล็ก หรือมากกว่าหนึ่งคลัสเตอร์ถ้าเป็นไฟล์ใหญ่ คำว่าคลัสเตอร์ดอสจะหมายถึงหน่วยที่ใช้จัดวางตำแหน่งของไฟล์ในดิสก์ที่ประกอบด้วยหนึ่งเซกเตอร์หรือหลายเซกเตอร์ขึ้นกับชนิดของดิสก์ เช่น ดิสก์ชนิดความจุหนาแน่นเป็น 2 เท่าแบบ 360 กิโลไบต์ จะใช้ 2 เซกเตอร์เป็น 1 คลัสเตอร์ เป็นต้น การฟอร์แมตดิสก์ขึ้นใหม่ก็จะมีการสร้างตารางหรือผังจัดวางไฟล์ (FAT) ซึ่งผังนี้จะบอกตำแหน่งว่างและข้อมูลคลัสเตอร์ภายในดิสก์ เวลาที่มีการสร้างไฟล์ขึ้น หมายเลขคลัสเตอร์ของไฟล์จะถูกเก็บบันทึกแบบเรียงต่อกันเป็นลูกโซ่ใน FAT อธิบายตำแหน่งที่เก็บจริงภายในดิสก์

การอ่านหรือเขียนไฟล์ทุกครั้ง ดอสต้องคำนวณหาตำแหน่งที่เก็บไฟล์ในดิสก์จากตาราง FAT ก่อนเสมอ เมื่อเรานำหมายเลขคลัสเตอร์ไฟล์มาเก็บในหน่วยความจำแทนแล้ว การทำงานติดต่อไฟล์จะรวดเร็วขึ้นอย่างมาก

จากที่ได้อธิบายในคำสั่ง FASTOPEN ที่ผ่านมา ข้อมูลคลัสเตอร์ของไฟล์หรือข้อมูล FAT จะถูกอ่านเข้ามาในบัฟเฟอร์ดิสก์ที่เป็นหน่วยความจำอยู่แล้ว เราอาจคิดว่าไม่จำเป็นต้องใช้วิธีการ Fast Seek เลย แต่จริง ๆ แล้วข้อมูลคลัสเตอร์ของไฟล์ที่เรียงติดต่อกันเป็นลูกโซ่อาจยาวข้ามระหว่างเซกเตอร์ของ FAT ทำให้ดอสต้องอ่านเซกเตอร์ของ FAT เข้ามาหลายเซกเตอร์ จึงเปลืองเนื้อที่ของบัฟเฟอร์ ขณะเดียวกันยังทำให้วิธีการ LRU ที่จัดการบัฟเฟอร์ดิสก์ทำงานไม่ถูกต้องด้วย ทำให้ระบบทั้งหมด

ทำงานล่าช้าเกินไป

ในดอส 4 ได้กำหนดรูปแบบคำสั่ง FASTOPEN เพิ่มเติมขึ้นอีกดังนี้

FASTOPEN d: = (n,m)

โดย d: เป็นช่องดิสก์ไดรฟ์ที่ต้องการ cache ข้อมูล

n เป็นจำนวนรายการไดเรคทอรีหรือไฟล์ กำหนดค่าได้จาก 10 ถึง 999

m เป็นจำนวนรายการคลัสเตอร์ที่เรียงต่อ (สำหรับ Fast Seek) กำหนดค่าได้จาก 1 ถึง 999

หรือ FASTOPEN d: /x

เป็นคำสั่งที่ให้ FASTOPEN ใช้หน่วยความจำส่วนเพิ่มขยาย (Expanded memory) ซึ่งเมื่อใช้สวิตช์ /x นี้กับ FASTOPEN แล้วจะต้องใช้สวิตช์นี้กับคำสั่ง BUFFERS ด้วย

แต่ละรายการคลัสเตอร์ที่เรียงต่อสำหรับ Fast Seek นั้นจะเก็บหมายเลขคลัสเตอร์ที่เริ่มต้นตามด้วยจำนวนคลัสเตอร์ที่เรียงต่อตามลำดับ ดังนั้นไฟล์ที่มีการเก็บบันทึกในดิสก์แบบเรียงต่อตามลำดับมากก็จะยิ่งใช้รายการคลัสเตอร์สำหรับ Fast Seek น้อยลง กรณีที่จำนวนรายการคลัสเตอร์กำหนดไว้ไม่พอ เพราะไฟล์ไม่ต่อเนื่องกันหรือมีไฟล์เปิดขึ้นพร้อม ๆ กันหลายไฟล์ Fast Seek ก็จะใช้วิธีการ LRU เพื่อปลดหรือยกเลิกค่าเก่าและแทนที่ด้วยค่าใหม่ทันที

คำสั่ง FASTOPEN เป็นคำสั่งที่น่าสนใจมากสำหรับผู้ใช้งานคอมพิวเตอร์ที่มีการจัดการระบบไฟล์เป็นโครงสร้างแบบต้นไม้จำนวนมาก ๆ เราอาจกำหนดคำสั่งนี้ในไฟล์อัตโนมัติของ AUTOEXEC.BAT หรืออาจกำหนดลงในคำสั่ง INSTALL ที่บรรจุในไฟล์ CONFIG.SYS เช่น

```
INSTALL = C:\DOS4\
FASTOPEN.EXE C:=(50,25)
```

สรุป

คำสั่ง BUFFERS และ FASTOPEN เป็นคำสั่งที่ได้รับการปรับปรุงเพิ่มขยายให้ช่วยงานด้านการเรียกติดต่อข้อมูลจากดิสก์ได้รวดเร็วขึ้น ด้วยการใช้หน่วยความจำมาเก็บข้อมูลที่สำคัญไว้ ในการเรียกใช้ข้อมูลครั้งต่อไปก็จะค้นจากในหน่วยความจำแทนที่จะต้องออกไปถึงดิสก์ซึ่งทำงานช้ากว่า ในดอส 4 ยังได้เพิ่มเติมคุณประโยชน์ให้สามารถใช้หน่วยความจำเพิ่มขยาย (Expanded Memory) มาเป็น Cache memory แทนหน่วยความจำปกติ เพื่อให้โปรแกรมประยุกต์ใช้หน่วยความจำปกติได้เต็มที่ เมื่อได้ติดตั้งทั้งคำสั่ง BUFFERS และ FASTOPEN ลงในระบบจะช่วยทำให้เพิ่มสมรรถภาพของการใช้งานได้สูงมากขึ้น

ทั้งคำสั่ง BUFFERS และ FASTOPEN เป็นคำสั่งที่ให้มาอยู่แล้วในดอส การทำความเข้าใจ รู้จักวิธีการใช้ที่ดีจะช่วยให้ระบบงานดีมีประสิทธิภาพยิ่งขึ้น ผู้เขียนขอแนะนำให้ผู้ใช้งานดอสได้ลองติดตั้ง เลือกกำหนดค่าบัฟเฟอร์ดิสก์ บัฟเฟอร์มองล่วงหน้า รายการไดเรคทอรีไฟล์ และรายการคลัสเตอร์ไฟล์ให้พอเหมาะกับระบบงานที่ใช้อยู่ แล้วเราจะพบว่าโปรแกรมประยุกต์ที่ใช้งานบนเครื่องเดียวกันนี้สามารถทำงานได้เร็วขึ้นอย่างมากทีเดียว

พบกันใหม่ฉบับหน้า...สวัสดีครับ

แก้ที่ผิดในไมโครฯ ฉบับที่ 50

ในคอลัมน์ "จดหมาย" หน้า 173 บรรทัดที่ 14

ข้อความเดิม "โดยอาศัยเนื้อที่บนฮาร์ดแวร์"

แก้เป็น "โดยอาศัยเนื้อที่บนฮาร์ดดิสก์"

ในคอลัมน์ "มุมภาษาซี" หน้า 315

ข้อความเดิม
```
C > masm-me tsrasm;
C > masm-me videoa;
```

แก้เป็น
```
C > masm-MI tsrasm;
C > masm-MI videoa;
```